

Reverse Berlekamp-Massey Decoding

Jiun-Hung Yu and Hans-Andrea Loeliger

Department of Information Technology and Electrical Engineering

ETH Zurich, Switzerland

Email: {yu, loeliger}@isi.ee.ethz.ch

Abstract—We propose a new algorithm for decoding Reed-Solomon codes (up to half the minimum distance). The proposed algorithm is similar in spirit and structure to the Berlekamp-Massey algorithm, but it applies to more general codes (including polynomial remainder codes). The algorithm can also be used to compute inverses in $F[x]/m(x)$.

I. INTRODUCTION

Let F be a finite field, let $\beta_0, \dots, \beta_{n-1}$ be n different elements of F , let $m(x) \triangleq \prod_{\ell=0}^{n-1} (x - \beta_\ell)$, let $F[x]/m(x)$ be the ring of polynomials modulo $m(x)$, and let ψ be the evaluation mapping

$$\psi : F[x]/m(x) \rightarrow F^n : a(x) \mapsto (a(\beta_0), \dots, a(\beta_{n-1})), \quad (1)$$

which is a ring isomorphism. A Reed-Solomon code [1], [2] with blocklength n and dimension k may be defined as

$$\{c = (c_0, \dots, c_n) \in F^n : \deg \psi^{-1}(c) < k\}, \quad (2)$$

usually with the additional condition that

$$\beta_\ell = \alpha^\ell \quad \text{for } \ell = 0, \dots, n-1, \quad (3)$$

where $\alpha \in F$ is a primitive n -th root of unity. The condition (3) implies

$$m(x) = x^n - 1 \quad (4)$$

and turns ψ into a discrete Fourier transform [3].

The standard algorithms for decoding Reed-Solomon codes up to half the minimum distance¹ are based either on the Berlekamp-Massey algorithm [4], [5] or on the Euclidean algorithm for computing the gcd (greatest common divisor) of two polynomials [6], [2], [3], [7]. For Berlekamp-Massey decoding, the condition (4) is essential.

In this paper, we propose an algorithm that is similar in spirit and structure to the Berlekamp-Massey algorithm, but it works for any code defined by (2) and does not need (3) and (4). Moreover, the proposed algorithm works also for polynomial remainder codes [9]–[12], which include Reed-Solomon codes as a special case. Nonetheless, in the special case (3) and (4), the algorithm is almost identical to (and as efficient as) the Berlekamp-Massey algorithm, except that it processes the syndrome polynomial in reverse order. However, the derivation of the algorithm is based on a new problem formulation (to be stated in Section II) that may be of interest in its own right. In particular, the proposed algorithm can also be used to compute

¹Decoding beyond half the minimum distance (cf. [8], [7]) is beyond the scope of this paper.

the inverse of $b(x)$ in $F[x]/f(x)$ (for any fixed $f(x) \in F[x]$ with $\deg f(x) \geq 1$) if it exists.

It should be mentioned here that Berlekamp-Massey decoding and gcd-based decoding are well known to be related, and explicit translations were given in [13], [14]. In fact, the algorithm proposed in this paper may be used as an intermediate in such a translation, which makes the translation more transparent. However, we will not elaborate this topic in the present paper.

The paper is structured as follows. The new problem formulation and the proposed algorithm are given in Section II. The application of the proposed algorithm to decoding Reed-Solomon codes is discussed in Section III. The proposed algorithm is explained and proved in Sections IV and V. The application of the proposed algorithm to decoding polynomial remainder codes is outlined in the appendix.

We will use the following notation. The Hamming weight of $e \in F^n$ will be denoted by $w_H(e)$. The coefficient of x^ℓ of a polynomial $b(x) \in F[x]$ will be denoted b_ℓ . The leading coefficient (i.e., the coefficient of $x^{\deg b(x)}$) of a nonzero polynomial $b(x)$ will be denoted by $\text{lcf } b(x)$, and we also define $\text{lcf}(0) \triangleq 0$. We will use “mod” both as in $r(x) = b(x) \bmod m(x)$ (the remainder of a division) and as in $b(x) \equiv r(x) \bmod m(x)$ (a congruence modulo $m(x)$).

II. PROBLEM STATEMENT AND PROPOSED ALGORITHM

As we shall see in Section III, decoding can be reduced to the following problem.

Problem: For given nonzero polynomials $b(x)$ and $m(x) \in F[x]$ with $\deg b(x) < \deg m(x)$, and given $d \in \mathbb{Z}$, $1 \leq d \leq \deg m(x)$, find a nonzero polynomial $\Lambda(x) \in F[x]$ of the smallest degree such that

$$\deg (b(x)\Lambda(x) \bmod m(x)) < d. \quad (5)$$

Remarks:

- 1) The stated assumptions imply $\deg m(x) \geq 1$.
- 2) In contrast to Section I, we do not assume here that $m(x)$ is a product of linear factors: any nonzero $m(x)$ is admitted.
- 3) For $d = \deg m(x)$, $\Lambda(x) = 1$ will do. Smaller values of d will normally require a polynomial $\Lambda(x)$ of higher degree.
- 4) Eq. (5) has always at least one solution $\Lambda(x)$ for any $d \geq 1$. If $\gcd(b(x), m(x)) = 1$, then $b(x)$ has an inverse in $F[x]/m(x)$; choosing $\Lambda(x)$ to be that inverse yields

□

$b(x)\Lambda(x) \bmod m(x) = 1$. If $\deg(\gcd(b(x), m(x))) > 0$, choosing $\Lambda(x) = m(x)/\gcd(b(x), m(x))$ yields $b(x)\Lambda(x) \bmod m(x) = 0$.

- 5) We will see that the solution $\Lambda(x)$ of the stated problem is unique up to a scale factor (Proposition 2 in Section IV) and satisfies

$$\deg \Lambda(x) \leq \deg m(x) - d \quad (6)$$

(by (47) in Section V).

- 6) Because of (6), the coefficients b_ℓ with

$$\ell < 2d - \deg m(x) \quad (7)$$

are irrelevant.

The stated problem is solved by the following algorithm.

Proposed Algorithm:

Input: $b(x)$, $m(x)$, and d as in the problem stated above.

Output: $\Lambda(x)$ as in the problem statement.

```

1  if  $\deg b(x) < d$  begin
2    return  $\Lambda(x) := 1$ 
3  end
4   $\Lambda^{(1)}(x) := 0$ ,  $d_1 := \deg m(x)$ ,  $\kappa_1 := \text{lcf } m(x)$ 
5   $\Lambda^{(2)}(x) := 1$ ,  $d_2 := \deg b(x)$ ,  $\kappa_2 := \text{lcf } b(x)$ 
6  loop begin
7     $\Lambda^{(1)}(x) := \kappa_2 \Lambda^{(1)}(x) - \kappa_1 x^{d_1 - d_2} \Lambda^{(2)}(x)$ 


---


8     $d_1 := \deg(b(x)\Lambda^{(1)}(x) \bmod m(x))$ 
9    if  $d_1 < d$  begin
10     return  $\Lambda(x) := \Lambda^{(1)}(x)$ 
11   end
12    $\kappa_1 := \text{lcf}(b(x)\Lambda^{(1)}(x) \bmod m(x))$ 


---


13   if  $d_1 < d_2$  begin
14      $(\Lambda^{(1)}(x), \Lambda^{(2)}(x)) := (\Lambda^{(2)}(x), \Lambda^{(1)}(x))$ 
15      $(d_1, d_2) := (d_2, d_1)$ 
16      $(\kappa_1, \kappa_2) := (\kappa_2, \kappa_1)$ 
17   end
18 end

```

Note that lines 14–16 simply swap $\Lambda^{(1)}(x)$ with $\Lambda^{(2)}(x)$, d_1 with d_2 , and κ_1 with κ_2 . The only actual computations are in lines 7 and 8.

The correctness of this algorithm will be proved in Section V. In particular, we will see that the value of d_1 is reduced in every execution of line 8.

Note that lines 8 and 12 do not require the computation of the entire polynomial $b(x)\Lambda^{(1)}(x) \bmod m(x)$. Indeed, lines 8–12 can be replaced by the following loop:

Equivalent Alternative to Lines 8–12:

```

31  repeat
32     $d_1 := d_1 - 1$ 
33    if  $d_1 < d$  begin
34      return  $\Lambda(x) := \Lambda^{(1)}(x)$ 
35    end

```

36 $\kappa_1 := \text{coefficient of } x^{d_1} \text{ in}$
 $b(x)\Lambda^{(1)}(x) \bmod m(x)$
 37 **until** $\kappa_1 \neq 0$

In the special case where $m(x) = x^n - 1$, line 36 amounts to

$$41 \quad \kappa_1 := b_{d_1} \Lambda_0^{(1)} + b_{[d_1-1]} \Lambda_1^{(1)} + \dots + b_{[d_1-\tau]} \Lambda_\tau^{(1)}$$

with $\tau \triangleq \deg \Lambda^{(1)}(x)$ and where the $b_{[\ell]} \triangleq b_{\ell \bmod n}$. In this case, the proposed algorithm looks very much like, and is as efficient as, the Berlekamp-Massey algorithm [5].

We conclude this section by noting that the proposed algorithm can also be used to compute the inverse of $b(x)$ in $F[x]/m(x)$, if it exists (cf. Remark 4 above). To this end, run the algorithm with $d = 1$. The algorithm then returns a polynomial $\Lambda(x)$ such that

$$b(x)\Lambda(x) \bmod m(x) \in F \setminus \{0\} \quad (8)$$

if an inverse exists and $b(x)\Lambda(x) \bmod m(x) = 0$ if no inverse exists.

III. APPLICATION OF THE PROPOSED ALGORITHM TO DECODING REED-SOLOMON CODES

Let \mathcal{C} be a (n, k) Reed-Solomon code over F as in Section I, but without requiring (3) and (4). Let $y = (y_0, \dots, y_{n-1}) \in F^n$ be the received word, which we wish to decompose into

$$y = c + e \quad (9)$$

where $c \in \mathcal{C}$ is a codeword and where the Hamming weight of $e = (e_0, \dots, e_{n-1}) \in F^n$ is as small as possible.

Let $C(x) \triangleq \psi^{-1}(c)$, and analogously $E(x) \triangleq \psi^{-1}(e)$ and $Y(x) \triangleq \psi^{-1}(y)$. Clearly, we have $\deg C(x) < k$ and $\deg E(x) < \deg m(x) = n$.

For any $e \in F^n$, we define (in the usual way) the error locator polynomial

$$\Lambda_e(x) \triangleq \prod_{\substack{\ell \in \{0, \dots, n-1\} \\ e_\ell \neq 0}} (x - \beta_\ell). \quad (10)$$

Clearly, $\deg \Lambda_e(x) = w_H(e)$ and

$$E(x)\Lambda_e(x) \bmod m(x) = 0. \quad (11)$$

Theorem 1. If $w_H(e) \leq \frac{n-k}{2}$, then the error locator polynomial $\Lambda_e(x)$ satisfies

$$\deg(Y(x)\Lambda_e(x) \bmod m(x)) < n - \frac{n-k}{2} \quad (12)$$

Conversely, for any y and $e \in F^n$ and $t \in \mathbb{R}$ with

$$w_H(e) \leq t \leq \frac{n-k}{2}, \quad (13)$$

if some nonzero $\Lambda(x) \in F[x]$ with $\deg \Lambda(x) \leq t$ satisfies

$$\deg(Y(x)\Lambda(x) \bmod m(x)) < n - t, \quad (14)$$

then $\Lambda(x)$ is a multiple of $\Lambda_e(x)$. \square

The proof is given below. We thus arrive at the following decoding procedure:

- 1) Compute $Y(x) = \psi^{-1}(y)$.
- 2) Run the algorithm of Section II with $b(x) = Y(x)$ and $d = \lceil \frac{n+k}{2} \rceil$. If $w_H(e) \leq \frac{n-k}{2}$, then the polynomial $\Lambda(x)$ returned by the algorithm equals $\Lambda_e(x)$ up to a scale factor.
- 3) Complete decoding by any standard method [3] or by means of Proposition 1 below.

Note that in Step 2, because of (7), coefficients Y_ℓ of $Y(x)$ with

$$\ell < 2 \left\lfloor \frac{n+k}{2} \right\rfloor - n \quad (15)$$

$$= \begin{cases} k, & \text{if } n-k \text{ is even} \\ k+1, & \text{if } n-k \text{ is odd} \end{cases} \quad (16)$$

are irrelevant.

As mentioned, decoding can be completed by the following proposition:

Proposition 1. If $\Lambda(x)$ is a nonzero multiple of $\Lambda_e(x)$ with $\deg \Lambda(x) \leq n-k$, then

$$C(x) = \frac{Y(x)\Lambda(x) \bmod m(x)}{\Lambda(x)} \quad (17)$$

□

Proof: If $\Lambda(x)$ has the stated properties, then

$$Y(x)\Lambda(x) \bmod m(x) = C(x)\Lambda(x) \bmod m(x) + E(x)\Lambda(x) \bmod m(x) \quad (18)$$

$$= C(x)\Lambda(x), \quad (19)$$

where the second term in (18) vanishes because of (11). □

Proof of Theorem 1: From (19), we have

$$\deg(Y(x)\Lambda_e(x) \bmod m(x)) < k + w_H(e), \quad (20)$$

and (12) follows from $k + w_H(e) \leq k + \frac{n-k}{2} = n - \frac{n-k}{2}$.

As for the converse, assume (13), (14), and $\deg \Lambda(x) \leq t$ and consider

$$Y(x)\Lambda(x) \bmod m(x) = C(x)\Lambda(x) + E(x)\Lambda(x) \bmod m(x). \quad (21)$$

Under the stated assumptions, the degree of the left-hand side of (21) is smaller than $n-t$ and also

$$\deg(C(x)\Lambda(x)) < k + t \leq n-t. \quad (22)$$

It follows that

$$\deg(E(x)\Lambda(x) \bmod m(x)) < n-t. \quad (23)$$

Now write

$$E(x)\Lambda(x) = Q(x)m(x) + E(x)\Lambda(x) \bmod m(x) \quad (24)$$

according to the polynomial division theorem. But $E(x)$ (and thus $E(x)\Lambda(x)$) has at least $n - w_H(e) \geq n-t$ zeros in the set $\{\beta_0, \beta_1, \dots, \beta_{n-1}\}$. It follows that $E(x)\Lambda(x) \bmod m(x)$ has also at least $n-t$ zeros (in this set), which contradicts (23) unless

$$E(x)\Lambda(x) \bmod m(x) = 0. \quad (25)$$

But any nonzero polynomial $\Lambda(x)$ that satisfies (25) is a multiple of the error locator polynomial (10). □

IV. KEY ELEMENTS OF THE PROOF

We return to the problem and the algorithm proposed in Section II. In this section, we discuss some key elements of the proposed algorithm and its proof. The actual proof will then be given in Section V.

The pivotal part of the algorithm is line 7, which is explained by the following lemma. (The corresponding statement for the Berlekamp-Massey algorithm is known as the *two-wrongs-make-a-right*² lemma.)

Lemma 1. Let $m(x)$ be a polynomial over F with $\deg m(x) \geq 1$. For further polynomials $b(x), \Lambda^{(1)}(x), \Lambda^{(2)}(x) \in F[x]$, let

$$r^{(1)}(x) \triangleq b(x)\Lambda^{(1)}(x) \bmod m(x), \quad (26)$$

$$r^{(2)}(x) \triangleq b(x)\Lambda^{(2)}(x) \bmod m(x), \quad (27)$$

$d_1 \triangleq \deg r^{(1)}(x)$, $\kappa_1 \triangleq \text{lcf } r^{(1)}(x)$, $d_2 \triangleq \deg r^{(2)}(x)$, $\kappa_2 \triangleq \text{lcf } r^{(2)}(x)$, and assume $d_1 \geq d_2 \geq 0$. Then

$$\Lambda(x) \triangleq \kappa_2 \Lambda^{(1)}(x) - \kappa_1 x^{d_1-d_2} \Lambda^{(2)}(x) \quad (28)$$

satisfies

$$\deg(b(x)\Lambda(x) \bmod m(x)) < d_1. \quad (29)$$

□

Proof: From (28), we obtain

$$r(x) \triangleq b(x)\Lambda(x) \bmod m(x) \quad (30)$$

$$= \kappa_2 r^{(1)}(x) - \kappa_1 x^{d_1-d_2} r^{(2)}(x) \quad (31)$$

by the natural ring homomorphism $F[x] \rightarrow F[x]/m(x)$. It is then obvious from (31) that $\deg r(x) < \deg r^{(1)}(x) = d_1$. □

A similar argument proves

Proposition 2 (Uniqueness of Solution). The solution $\Lambda(x)$ of the problem of Section II is unique (up to a scale factor). □

Proof: Let $\Lambda^{(1)}(x)$ and $\Lambda^{(2)}(x)$ be two solutions of the problem, which implies $\deg \Lambda^{(1)}(x) = \deg \Lambda^{(2)}(x) \geq 0$. Define $r^{(1)}(x)$ and $r^{(2)}(x)$ as in (26) and (27) and consider

$$\Lambda(x) \triangleq (\text{lcf } \Lambda^{(2)}(x)) \Lambda^{(1)}(x) - (\text{lcf } \Lambda^{(1)}(x)) \Lambda^{(2)}(x). \quad (32)$$

Then

$$r(x) \triangleq b(x)\Lambda(x) \bmod m(x) \quad (33)$$

$$= (\text{lcf } \Lambda^{(2)}(x)) r^{(1)}(x) - (\text{lcf } \Lambda^{(1)}(x)) r^{(2)}(x), \quad (34)$$

which implies that $\Lambda(x)$ also satisfies (5). But (32) implies $\deg \Lambda(x) < \deg \Lambda^{(1)}(x)$, which is a contradiction unless $\Lambda(x) = 0$. Thus $\Lambda(x) = 0$, which means that $\Lambda^{(1)}(x)$ and $\Lambda^{(2)}(x)$ are equal up to a scale factor. □

Definition (Remainder-Minimal): For fixed nonzero $b(x)$ and $m(x) \in F[x]$ with $\deg b(x) < \deg m(x)$ (as in the problem of

²So called by J. L. Massey in many of his lectures.

Section II), a nonzero polynomial $\Lambda(x) \in F[x]$ is *remainder-minimal* if

$$\deg(b(x)\Lambda^{(1)}(x) \bmod m(x)) \leq \deg(b(x)\Lambda(x) \bmod m(x)) \quad (35)$$

(with $\Lambda^{(1)}(x) \neq 0$) implies $\deg \Lambda^{(1)}(x) \geq \deg \Lambda(x)$.

The following lemma is the counterpart to Theorem 1 of [5].

Lemma 2 (Degree Change Lemma). For fixed nonzero $b(x)$ and $m(x) \in F[x]$ with $\deg b(x) < \deg m(x)$, let $\Lambda(x)$ be a remainder-minimal polynomial and let

$$r(x) \triangleq b(x)\Lambda(x) \bmod m(x). \quad (36)$$

If

$$\deg \Lambda(x) \leq \deg m(x) - \deg r(x), \quad (37)$$

then any nonzero polynomial $\Lambda^{(1)}(x) \in F[x]$ such that

$$\deg(b(x)\Lambda^{(1)}(x) \bmod m(x)) < \deg r(x) \quad (38)$$

satisfies

$$\deg \Lambda^{(1)}(x) \geq \deg m(x) - \deg r(x). \quad (39)$$

□

The proof is given below.

Corollary: Assume everything as in Lemma 2 including (37) and (38). If (39) is satisfied with equality, then $\Lambda^{(1)}(x)$ is also remainder-minimal.

Proof of Lemma 2: Assume that $\Lambda^{(1)}(x)$ is a nonzero polynomial that satisfies (38), i.e., the degree of

$$r^{(1)}(x) \triangleq b(x)\Lambda^{(1)}(x) \bmod m(x) \quad (40)$$

satisfies

$$\deg r^{(1)}(x) < \deg r(x). \quad (41)$$

Multiplying (36) by $\Lambda^{(1)}(x)$ and (40) by $\Lambda(x)$ yields

$$\Lambda^{(1)}(x)r(x) \equiv \Lambda(x)r^{(1)}(x) \bmod m(x). \quad (42)$$

If we assume both (37) and (contrary to (39))

$$\deg \Lambda^{(1)}(x) < \deg m(x) - \deg r(x), \quad (43)$$

then (42) reduces to

$$\Lambda^{(1)}(x)r(x) = \Lambda(x)r^{(1)}(x). \quad (44)$$

But then (41) implies $\deg \Lambda^{(1)}(x) < \deg \Lambda(x)$, which is impossible because $\Lambda(x)$ is remainder-minimal. Thus (37) and (43) cannot hold simultaneously. □

V. PROOF OF THE PROPOSED ALGORITHM

We now prove the correctness of the algorithm proposed in Section II. To this end, we restate the algorithm with added assertions as follows.

Proposed Algorithm Restated:

```

1  if  $\deg b(x) < d$  begin
2    return  $\Lambda(x) := 1$ 
3  end
4   $\Lambda^{(1)}(x) := 0, d_1 := \deg m(x), \kappa_1 := \text{lcf } m(x)$ 
5   $\Lambda^{(2)}(x) := 1, d_2 := \deg b(x), \kappa_2 := \text{lcf } b(x)$ 
6  loop begin
    Assertions:
     $d_1 > d_2 \geq d$  (A.1)
     $\deg \Lambda^{(2)}(x) = \deg m(x) - d_1$  (A.2)
     $> \deg \Lambda^{(1)}(x)$  (A.3)
     $\Lambda^{(2)}(x)$  is remainder-minimal (A.4)
7  repeat
8     $\Lambda^{(1)}(x) := \kappa_2 \Lambda^{(1)}(x) - \kappa_1 x^{d_1-d_2} \Lambda^{(2)}(x)$ 
    Assertions:
     $\deg(b(x)\Lambda^{(1)}(x) \bmod m(x)) < d_1$  (A.5)
     $\deg \Lambda^{(1)}(x) = \deg m(x) - d_2$  (A.6)
     $> \deg \Lambda^{(2)}(x)$  (A.7)
9     $d_1 := \deg(b(x)\Lambda^{(1)}(x) \bmod m(x))$ 
10   if  $d_1 < d$  begin
    Assertion:
     $\Lambda^{(1)}(x)$  is remainder-minimal (A.8)
    return  $\Lambda(x) := \Lambda^{(1)}(x)$ 
11   end
12    $\kappa_1 := \text{lcf}(b(x)\Lambda^{(1)}(x) \bmod m(x))$ 
13   until  $d_1 < d_2$ 
    Assertion:
     $\Lambda^{(1)}(x)$  is remainder-minimal (A.9)
14    $(\Lambda^{(1)}(x), \Lambda^{(2)}(x)) := (\Lambda^{(2)}(x), \Lambda^{(1)}(x))$ 
15    $(d_1, d_2) := (d_2, d_1)$ 
16    $(\kappa_1, \kappa_2) := (\kappa_2, \kappa_1)$ 
17 end

```

Note the added inner **repeat** loop (lines 7–14), which does not change the algorithm but helps to state its proof.

Throughout the algorithm (except at the very beginning, before the first execution of lines 9 and 13), d_1 , d_2 , κ_1 , and κ_2 are defined as in Lemma 1, i.e., $d_1 = \deg r^{(1)}(x)$, $\kappa_1 = \text{lcf } r^{(1)}(x)$, $d_2 = \deg r^{(2)}(x)$, and $\kappa_2 = \text{lcf } r^{(2)}(x)$ for $r^{(1)}(x)$ and $r^{(2)}(x)$ as in (26) and (27).

Assertions (A.1)–(A.4) are easily verified, both from the initialization and from (A.6), (A.7), and (A.9).

As for (A.5), after the very first execution of line 8, we still have $d_1 = \deg m(x)$ (from line 4), which makes (A.5) obvious. For all later executions of line 8, (A.5) follows from Lemma 1.

As for (A.6) and (A.7), we note that line 8 changes the degree of $\Lambda^{(1)}(x)$ as follows:

- Upon entering the **repeat** loop, line 8 increases the degree of $\Lambda^{(1)}$ to

$$\deg \Lambda^{(2)}(x) + d_1 - d_2 = \deg m(x) - d_2 \quad (45)$$

$$> \deg \Lambda^{(2)}(x), \quad (46)$$

which follows from (A.1)–(A.3).

- Subsequent executions of line 8 without leaving the **repeat** loop (i.e., without executing lines 15–17) do not change the degree of $\Lambda^{(1)}(x)$. (This follows from the fact that d_1 is smaller than in the first execution while $\Lambda^{(2)}(x)$, d_2 , and $\kappa_2 \neq 0$ remain unchanged.)

Assertion (A.9) follows from the Corollary to Lemma 2 (with $\Lambda(x) = \Lambda^{(2)}(x)$ and $\deg r(x) = d_2$), which applies because $d_1 < d_2$ and (A.6). Because of (A.1), the same argument applies also to (A.8).

Finally, (A.1) and (A.6) imply that the polynomial $\Lambda(x)$ returned by the algorithm satisfies

$$\deg \Lambda(x) \leq \deg m(x) - d. \quad (47)$$

VI. CONCLUSION

We have shown that decoding Reed-Solomon codes can be reduced to the problem stated in Section II, and we proposed a new algorithm for solving this problem. In the special case where $m(x) = x^n - 1$, the proposed algorithm almost coincides with the Berlekamp-Massey algorithm, except that it processes the syndrome in reverse order. However, the algorithm works for general $m(x)$ and even for polynomial remainder codes (cf. the appendix), and it can also be used to compute inverses in $F[x]/m(x)$.

APPENDIX: EXTENSION TO POLYNOMIAL REMAINDER CODES

The algorithm proposed in Section II can also be used to decode polynomial remainder codes [9]–[12], which include Reed-Solomon codes as a special case. (The Berlekamp-Massey algorithm does not apply to such codes.)

Let $m_0(x), \dots, m_{n-1}(x) \in F[x]$ be relatively prime and let $m(x) \triangleq \prod_{\ell=0}^{n-1} m_\ell(x)$. Let $R_m \triangleq F[x]/m(x)$ denote the ring of polynomials modulo $m(x)$ and let $R_{m_\ell} \triangleq F[x]/m_\ell(x)$. The mapping (1) is generalized to the ring isomorphism

$$\begin{aligned} \psi : R_m &\rightarrow R_{m_0} \times \dots \times R_{m_{n-1}} : \\ a(x) &\mapsto \psi(a) \triangleq (\psi_0(a), \dots, \psi_{n-1}(a)) \end{aligned} \quad (48)$$

with $\psi_\ell(a) \triangleq a(x) \bmod m_\ell(x)$. Following [12], a polynomial remainder code may be defined as

$$\{c = (c_0, \dots, c_{n-1}) \in R_{m_0} \times \dots \times R_{m_{n-1}} : \deg \psi^{-1}(c) < K\} \quad (49)$$

where

$$K \triangleq \sum_{\ell=0}^{k-1} \deg m_\ell(x) \quad (50)$$

for some fixed k , $0 < k < n$. We also define

$$N \triangleq \deg m(x) = \sum_{\ell=0}^{n-1} \deg m_\ell(x). \quad (51)$$

As in Section III, let $y = c + e$ be the received word with $c \in \mathcal{C}$, and let $C(x) \triangleq \psi^{-1}(c)$, $E(x) \triangleq \psi^{-1}(e)$, and $Y(x) \triangleq \psi^{-1}(y)$. Clearly, $\deg C(x) < K$ and $\deg E(x) < N$.

For such codes, the error locator polynomial

$$\Lambda_e(x) \triangleq \prod_{\substack{\ell \in \{0, \dots, n-1\} \\ e_\ell \neq 0}} m_\ell(x) \quad (52)$$

and the error factor polynomial [12]

$$\Lambda_f(x) \triangleq m(x) / \gcd(E(x), m(x)) \quad (53)$$

do not, in general, coincide. However, if all moduli $m_\ell(x)$ are irreducible, then $\Lambda_f(x) = \Lambda_e(x)$.

We then have the following generalization of Theorem 1:

Theorem 2. For given y and e with $\deg \Lambda_f(x) \leq t \leq \frac{N-K}{2}$, assume that some nonzero polynomial $\Lambda(x)$ with $\deg \Lambda(x) \leq t$ satisfies

$$\deg (Y(x)\Lambda(x) \bmod m(x)) < N - t. \quad (54)$$

Then $\Lambda(x)$ is a multiple of $\Lambda_f(x)$. Conversely, $\Lambda(x) = \Lambda_f(x)$ is a polynomial of the smallest degree that satisfies (54). \square

It follows that the decoding procedure of Section III works also for polynomial remainder codes, except that n , k , and $\Lambda_e(x)$ are replaced by N , K , and $\Lambda_f(x)$, respectively. Moreover, $C(x)$ can still be recovered from $\Lambda(x)$ by means of (17) [12].

REFERENCES

- [1] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *J. SIAM*, vol. 8, pp. 300–304, Oct. 1962.
- [2] J. Justesen and T. Høholdt, *A Course in Error Correcting Codes*. Europ. Math. Soc., 2004.
- [3] R. E. Blahut, "Algebraic Codes for Data Transmission." Cambridge University Press, Cambridge, UK, 2003.
- [4] E. R. Berlekamp, "Algebraic Coding Theory." New York: McGraw-Hill, 1968.
- [5] J. L. Massey, "Shift-register synthesis and BCH decoding," *IEEE Trans. Information Theory*, vol. 15, pp. 122–127, May 1969.
- [6] Y. Sugiyama, M. Kasahara, S. Hirasawa, and T. Namekawa, "A method for solving key equation for decoding Goppa codes," *Information and Control*, vol. 27, pp. 87–99, 1975.
- [7] R. M. Roth, *Introduction to Coding Theory*. New York: Cambridge University Press, 2006.
- [8] V. Guruswami and M. Sudan, "Improved decoding of Reed-Solomon codes and algebraic-geometric codes," *IEEE Trans. Information Theory*, vol. 45, pp. 1757–1767, Sept. 1999.
- [9] J. J. Stone, "Multiple-burst error correction with the Chinese Remainder Theorem," *J. SIAM*, vol. 11, pp. 74–81, Mar. 1963.
- [10] A. Shiozaki, "Decoding of redundant residue polynomial codes using Euclid's algorithm," *IEEE Trans. Information Theory*, vol. 34, pp. 1351–1354, Sep. 1988.
- [11] J.-H. Yu and H.-A. Loeliger, "On irreducible polynomial remainder codes," *IEEE Int. Symp. on Information Theory, Saint Petersburg, Russia, July 31–Aug. 5, 2011*.
- [12] J.-H. Yu and H.-A. Loeliger, "On polynomial remainder codes," <http://arxiv.org/abs/1201.1812>.
- [13] J. L. Dornstetter, "On the equivalence between Berlekamp's and Euclid's algorithms," *IEEE Trans. Information Theory*, vol. 33, pp. 428–431, May 1987.
- [14] A. E. Heydtmann and J. M. Jensen, "On the equivalence of the Berlekamp-Massey and Euclidean algorithms for decoding," *IEEE Trans. Information Theory*, vol. 46, pp. 2614–2624, Nov. 2000.